Monterey Bay Aquarium
Research Institute

Developing a Simulator for Collaborating LRAUVs

Ethan Park, Northwestern University

Mentors: Yanwu Zhang, Ben Yair Raanan

Summer 2019

Keywords: LRAUV, ROS, Gazebo, simulation

ABSTRACT

Since their inception over a decade ago, MBARI's Long-Range Autonomous Underwater Vehicles (LRAUV) have helped oceanographic researchers gather datasets ranging thousands of kilometers over weeks or months at a time. Critical to their success has been the ability to simulate these sophisticated robots before deployment. The custom simulator currently in use is far from user-friendly, but it is utilitarian and accurately shows an LRAUV undertaking its programmed mission. However, now that many LRAUV are in operation and with ambitious multi-LRAUV missions foreseeable for future research, the current simulator is insufficient for various reasons. The Unmanned Underwater Vehicle Simulator (UUV Simulator), a European research prototype created for use with the Robot Operating System (ROS), has been selected as a suitable candidate to meet this impending simulation need. This paper showcases the capabilities of the UUV Simulator and ROS by simulating two multi-LRAUV scenarios inspired by MBARI research and incorporating the LRAUV's dynamics from the current simulator into an existing UUV Simulator robot model.

INTRODUCTION

With the ability to virtually recreate actions or reconstruct entire landscapes, it is no surprise that simulations are some of the most powerful tools of the trade in any research field. They allow users to iterate scenarios thousands or even millions of times to train machine learning algorithms. They give us glimpses of ancient cities lost to time and locations that are light years away in deep space, places otherwise inaccessible to modern researchers. However, another important advantage that a simulation offers is the availability of a risk-free testbed. This is especially useful in robotics, because deploying or testing a robot in the real world is often extremely time and resource intensive, and a robot failure in the real world, catastrophic.

Researchers at the Monterey Bay Aquarium Research Institute (MBARI) know this fact all too well, due to tightly constrained ship days with which to conduct experiments, collect data, or deploy Autonomous Underwater Vehicles (AUV)s or Remotely Operated Vehicles (ROV)s, and the need to coordinate the complete presence of an often multi-institutional research team, along with all of the necessary equipment. Therefore, in the context of marine robots such as those used by MBARI, a robot failure can mean the total loss of the robot, leading to tremendous financial loss but more importantly, valuable research, engineering, and fabrication time lost. With a simulation, however, there is no risk to the robot. It is also often faster to incorporate bug fixes and other modifications and rerun the robot in a simulation than it is in the real world. Of course, in order to reap these benefits, a simulation must be similar enough to the real performance so that running the robot in simulation is a meaningful endeavor.

Figure 1: LRAUV Galene. Photo by Madison Heard, MBARI 2019

For this project, the robot(s) of interest is MBARI's Long-Range Autonomous Underwater Vehicle (LRAUV). Development of the LRAUV began in 2007, with the goal of supporting sensory

missions covering ranges in the thousands of kilometers (Monterey Bay Aquarium Research Institute, 2018). The LRAUV's endurance gives it valuable temporal flexibility, allowing it to wait for certain biological events to occur, or to respond to spontaneous events. MBARI now operates a fleet of LRAUVs, and besides in Monterey Bay, they have been deployed in Hawaii and the Great Lakes.

The LRAUV simulator in use currently is basically a live system log, a program that prints out every action, status, and error message of the LRAUV to a standard command prompt/terminal, thereby allowing the user to piece together what the LRAUV has done and is currently doing by reading the steady stream of messages printed to the terminal.

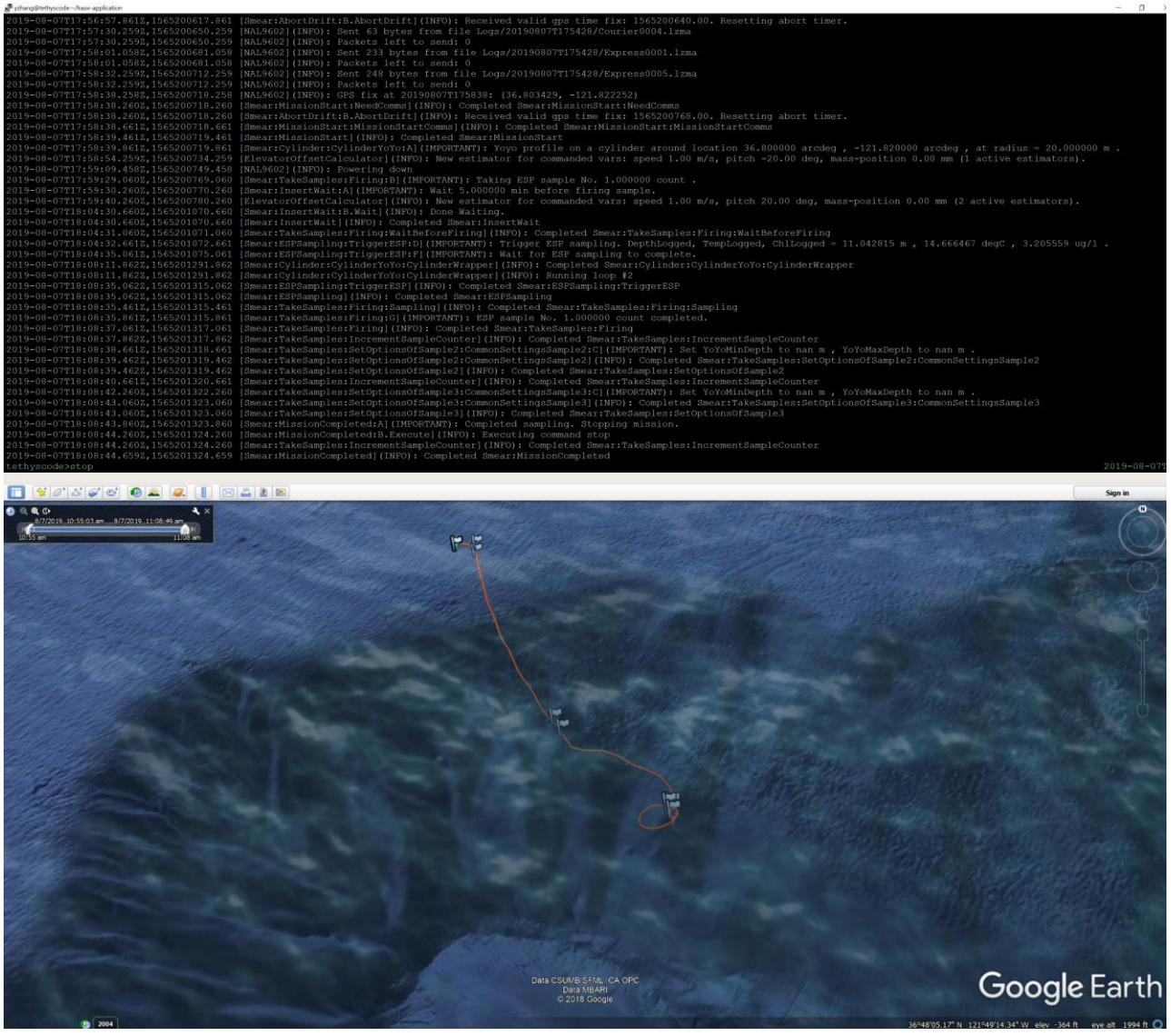

Figure 2: Current LRAUV Simulator, showing terminal view and trajectory uploaded to Google Earth. Image courtesy of Yanwu Zhang

There is no live visual of the LRAUV executing its mission, only an output *.kml* file that can be uploaded to Google Earth to see the path taken by the LRAUV. For those who are accustomed to the simulator or are familiar with the LRAUV's operational code, this simulator is sufficient. It portrays all the necessary information to determine whether the LRAUV has successfully ran its programmed mission or not.

However, consider a mission with multiple LRAUVs which are communicating with each other and perhaps even influencing each other's actions over a wide operational area. (In short, collaborating LRAUVs) Such multi-AUV missions are a reality today, and are very likely to be used in future oceanographic research.

In a mission like this, because there are multiple, collaborative robots involved, it is vital to be able to watch them as the mission unfolds. The success of the mission depends not just on the robots moving to the correct locations but also doing so while responding correctly to the behavior and information relayed live from their counterparts, which is best confirmed with a live visual. With what is available with the current simulator, even if an experienced user can piece together the robots' trajectories as a function of time, this painstaking method quickly becomes implausible with larger numbers of robots.

Furthermore, although the simulator implements acoustic pinging as tracking between different vehicles or platforms, there is currently no inter-vehicle messaging capability. Since sophisticated communication between the robots is what makes these multi-robot scenarios possible, the absence of this simulation capability is critical.

Lastly, with the current simulator, simulation of the ocean environment is limited. Environmental variables such as temperature and bathymetry are implemented, but more complex features such as chlorophyll patches and oil spills are difficult to simulate, exacerbated by the lack of a live visual interface. This is important because environmental features like these are often what necessitate multi-LRAUV operations, meaning that if these cannot be simulated well or even at all, then it is impossible to gain a realistic simulation of the LRAUVs. Thus, for all of the aforementioned reasons, the current LRAUV simulator is unsuitable as a simulator for scenarios with collaborating LRAUVs.

Figure 3: A patch of biodegradable dye like this, being tracked by the LRAUV to the far right, would be difficult to simulate with the current simulator. Image by Todd Walsh, MBARI 2018.

MATERIALS AND METHODS

Owing to the shortcomings of the current simulator in multi-LRAUV simulation, the LRAUV team has been looking for another simulation tool to fill this gap, namely ROS with the UUV Simulator, which is discussed below.

Robot Operating System (ROS)

Contrary to what the name suggests, ROS is not an operating system in the computer science sense of the term. Rather, it can be loosely described as a cluster of computing processes with a structured communications regime. ROS was conceived at Stanford University and Willow Garage, where students and researchers saw the need for an accessible and versatile robotics software framework upon noting the limitations of individual expertise and knowledge on a topically broad field like robotics. The design principles of ROS are as follows:

- a peer-to-peer network topology between many different processes often located on different hosts, networks, or machines;

- a microkernel design that relies on many small tools to build and run different ROS components, rather than one large module handling everything;
- multilingual support, due to individual programming language preferences;
- isolation of drivers and algorithms in standalone libraries for convenient reuse;
- free and open-source to facilitate debugging at all levels of the software stack.

(Quigley, et al., 2009).

Fundamentally, an instance of ROS is comprised of many computing processes, called *nodes* in ROS terminology. A node can take many forms, from all of the operational software within a robot, to just one small component, or maybe a behavioral algorithm running on a separate script. Communication between nodes takes place with messages sent via *topics* and *services*.

Topics are basically asynchronous message channels with a set message format. Any node that is a *publisher*, or sender, to a specific topic can send messages on that topic, and any *subscriber* node, or reader, to the same topic can read any message published on that topic. A node can also publish or subscribe to as many topics as need be. However, this method is one-directional; unless the publisher is explicitly distinguished within the information contained in a message (by an ID, flag, etc.), a subscriber has no way to identify the publisher of a particular message.

Though topics and their publish/subscribe communication scheme is useful, there are occasions where a continuous stream of messages is undesirable and a request/reply scheme is necessary. This is where *services* are used. Services are the synchronous form of ROS communication and are quite similar to general-purpose functions in programming. Requests and replies for a service must follow the set request or reply message format, similar to how functions in certain languages designate the output type and require certain inputs in order to execute correctly.

Lastly, despite the earlier statement that ROS differed from conventional operating systems, using ROS is in a way similar to using conventional operating systems like Windows or Mac. For a computer, the user finds a program or application online, installs it, and runs it. In ROS, programs and applications are akin to *packages*; the user can often

find the necessary ROS packages online, and therefore only need to install it (and its dependencies if necessary) and run it through ROS.

All in all, through a decade of development, ROS has become the workhorse of robotics research, with the original ROS paper having been cited over 5,800 times according to Google Scholar (Google, 2019), and seeing use by NASA for the Robonaut2 and by teams in the DARPA Robotics Challenge.

Simulation in ROS

A standard installation of ROS includes two programs called Gazebo and Rviz. These are used for simulation and visualization, respectively. Gazebo is akin to the common perception of 'simulation', complete with a physics engine, life-like models, and an environment/world. As a visualization tool, Rviz is a sort of puppet to the source of data that it displays, whether that be a simulation in Gazebo, or a robot in the real world. Therefore, a user can forego Rviz entirely, if desired. The figure below illustrates this relationship.

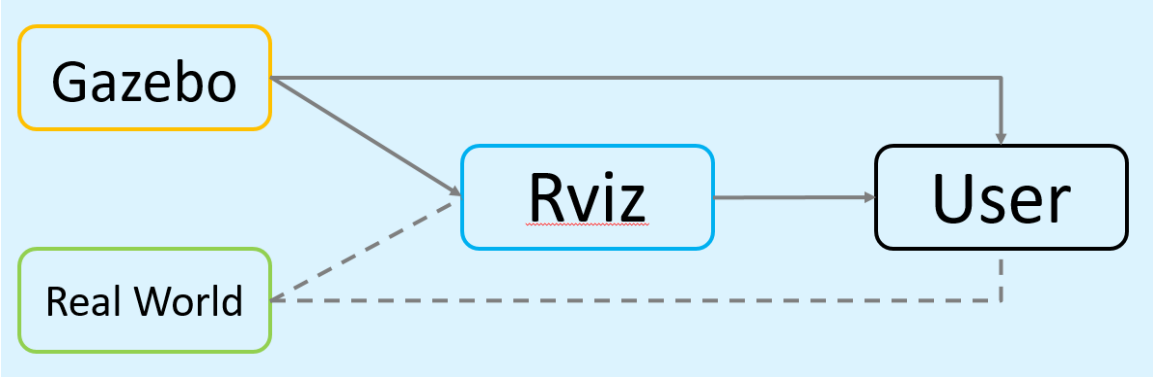

Figure 4: ROS Simulation flowchart.

Unmanned Underwater Vehicle Simulator (UUV Simulator)

As it comes, ROS is ready to simulate terrestrial robots and worlds. However, for underwater robots and marine environments, additional packages are necessary. This is where the UUV Simulator comes into play. Developed as a research prototype for the EU ECSEL Project 662107 SWARMS, UUV Simulator implements additional Gazebo

plugins and ROS nodes necessary for realistic underwater vehicle simulation, such as for ROVs and AUVs. These include:

- implementation of Fossen's equations of motion for underwater vehicles (detailed further in paper)
- thruster modules to translate thruster angular velocity to thrust force
- lift and drag for fins
- three-dimensional current
- underwater vehicle sensors
- a variety of controllers
- teleoperation of AUVs and ROVs
- ocean worlds taken from real settings (Romania, Norway)
- five sample vehicle models

(M. M. M. Manhães et al, 2016)

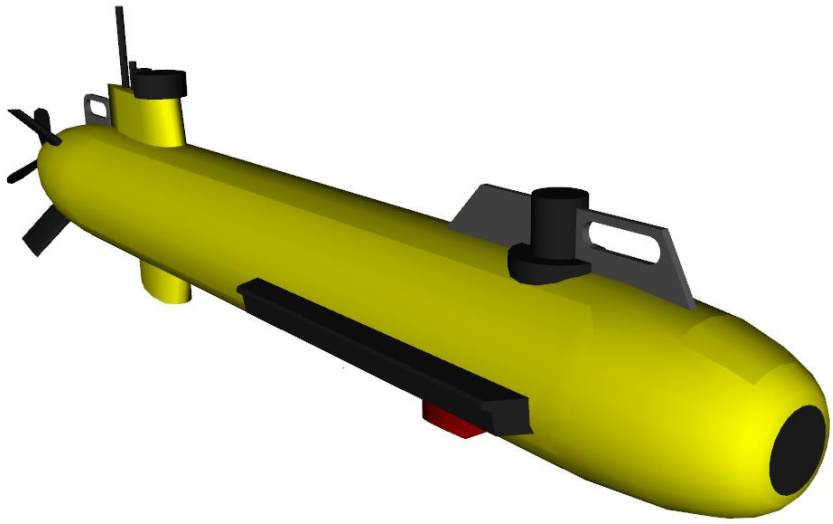

Figure 5: ECA Group's A9 AUV, modelled in UUV Simulator. This model was used extensively to represent the LRAUV for this project. Image source: UUV Simulator.

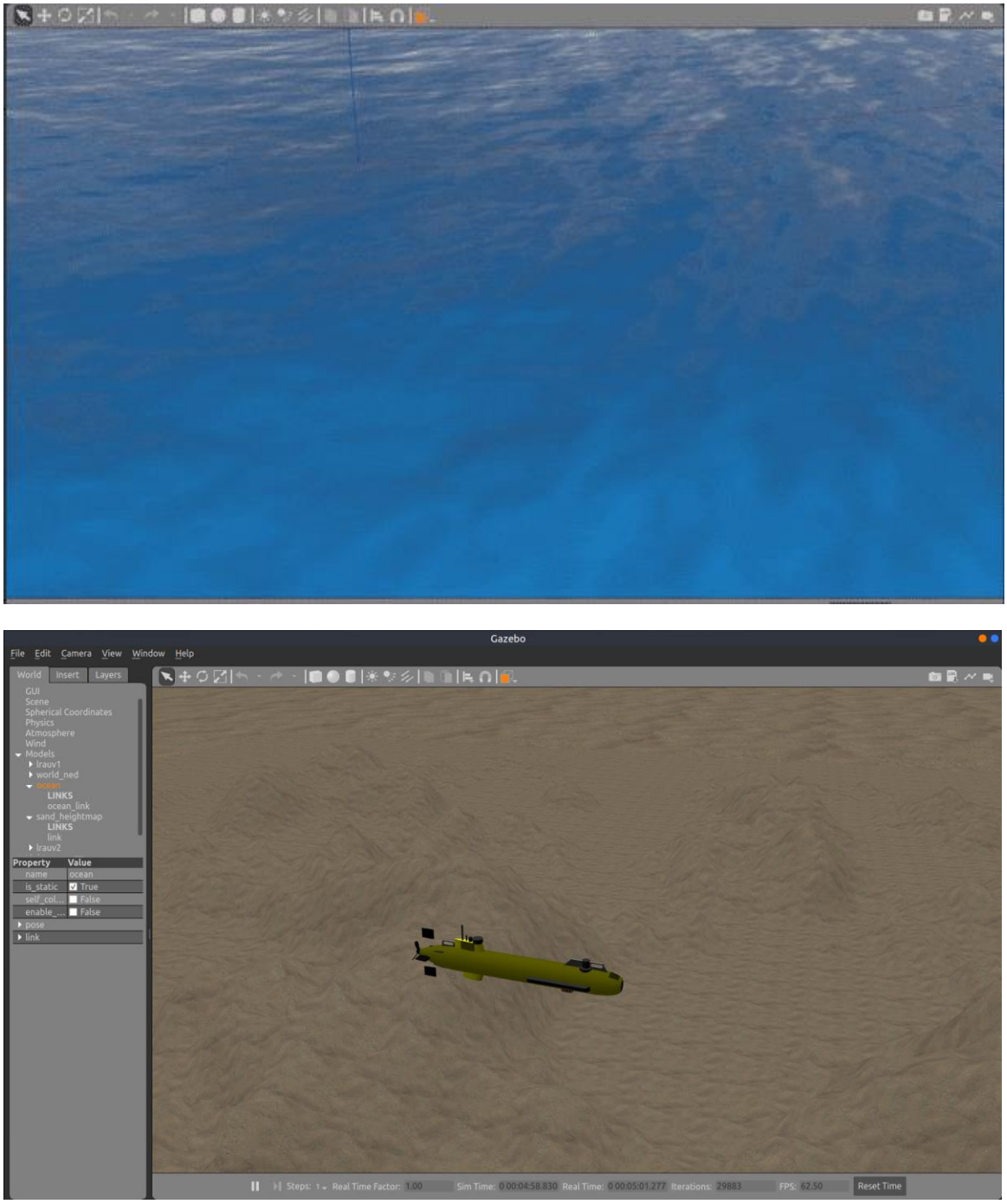

Figure 6: Sample Gazebo world of UUV Simulator showing modelled ocean surface, ECA A9 model underwater, and realistic seabed.

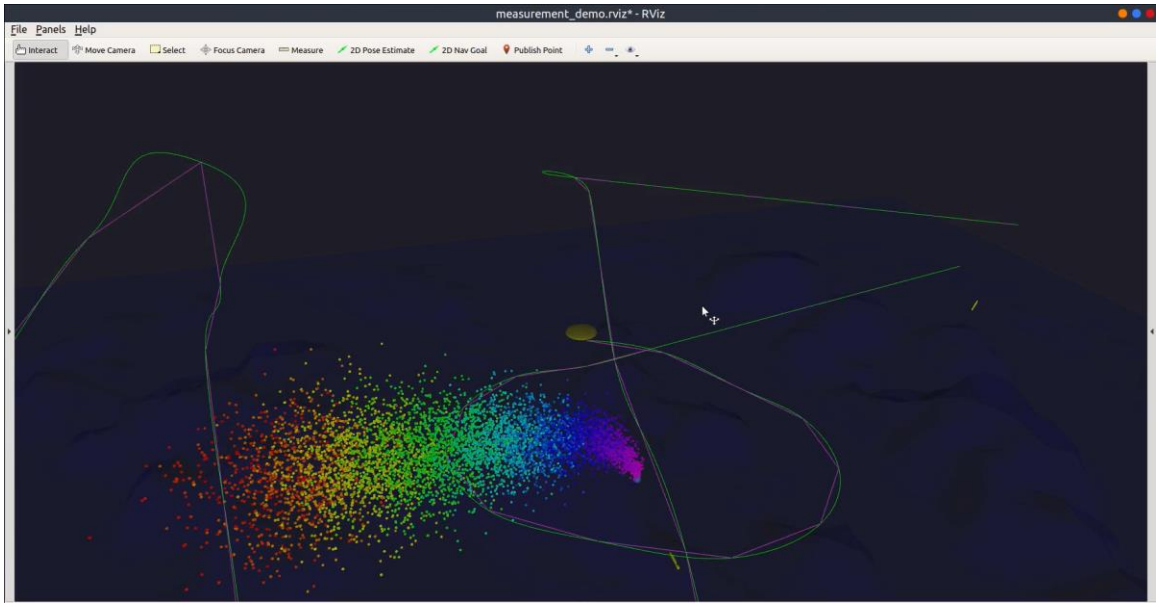

Figure 7: UUV Simulator in Rviz, showing some AUV waypoints, trajectories, and particle plume, visualized, alongside the AUVs.

To evaluate the capability of the UUV Simulator, two scenarios drawn from MBARI's own research were devised and simulated. These were: a recreation of certain elements of the 2019 Spring CANON experiment, and oceanographic hotspot detection using targeted sampling.

Scenario 1 – CANON (Principal Investigator: Francisco Chavez)

Beginning in late May, MBARI researchers began a week-long, "Controlled, Agile, and Novel Observing Network (CANON)" experiment, to study the diurnal vertical migration of animals such as copepods, krill, and fish. Among the array of technology used for this experiment were the LRAUVs, which carried MBARI's custom made Environmental Sample Processors to collect environmental DNA, eDNA for short (Brown, 2019). From the experiment, the following scene was taken as inspiration for simulation. Within this image, the two LRAUVs maneuvering around the Wave Glider in the center – the area for which is delineated with the yellow circle – was programmed into the UUV Simulator.

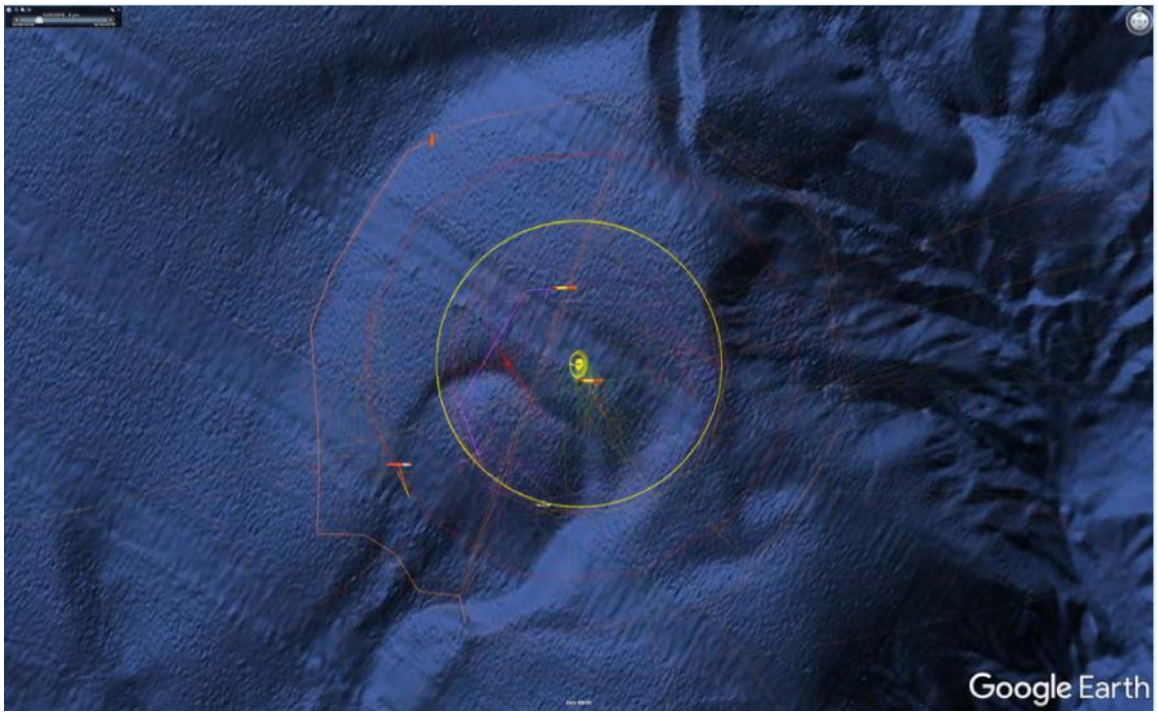

Figure 8: CANON experiment LRAUV maneuvers. Source: Kevin Gomes.

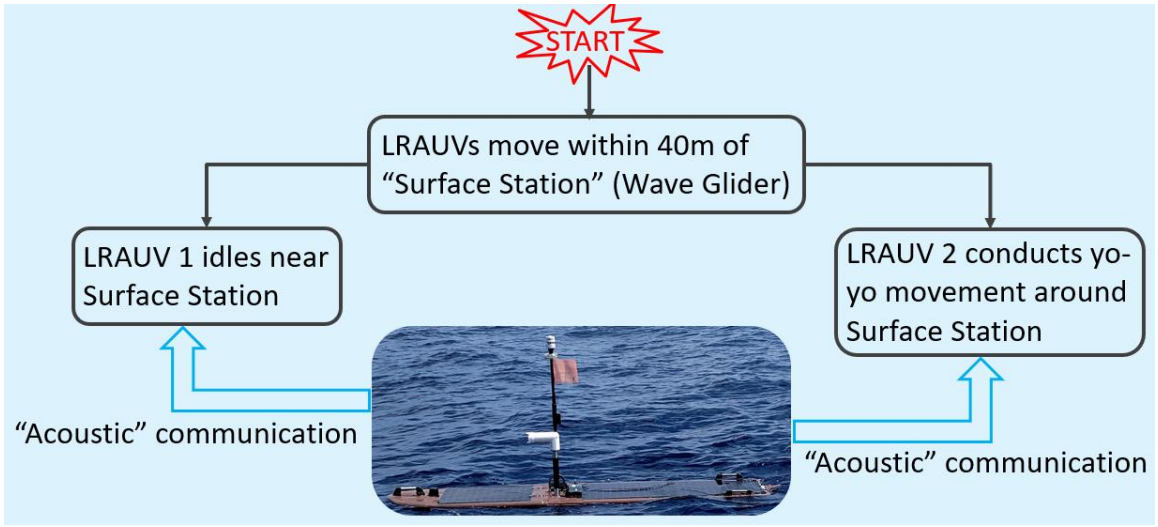

Figure 9: Breakdown of CANON-inspired simulation in UUV Simulator.

To explain the diagram above, there are three main components in this scenario: LRAUV 1, LRAUV 2, and the "Surface Station", which represents the Wave Glider from the CANON image above. The Surface Station remains stationary in one position for the entirety of the simulation, while the two LRAUVs start out some distance away, far enough where the first behavior that they will engage in is to simply close the distance between them and the Surface Station. Once they are within 40 meters, LRAUV 1 will proceed to the Surface Station's position and idle there, while LRAUV 2 will engage in a

roughly circular gait around the Surface Station, all the while also performing a mild yo-yo (vertical zigzagging, done for research purposes to sample throughout the water column) movement. Throughout the simulation, both LRAUVs are in constant communication with the Surface Station, the schema for which is shown below.

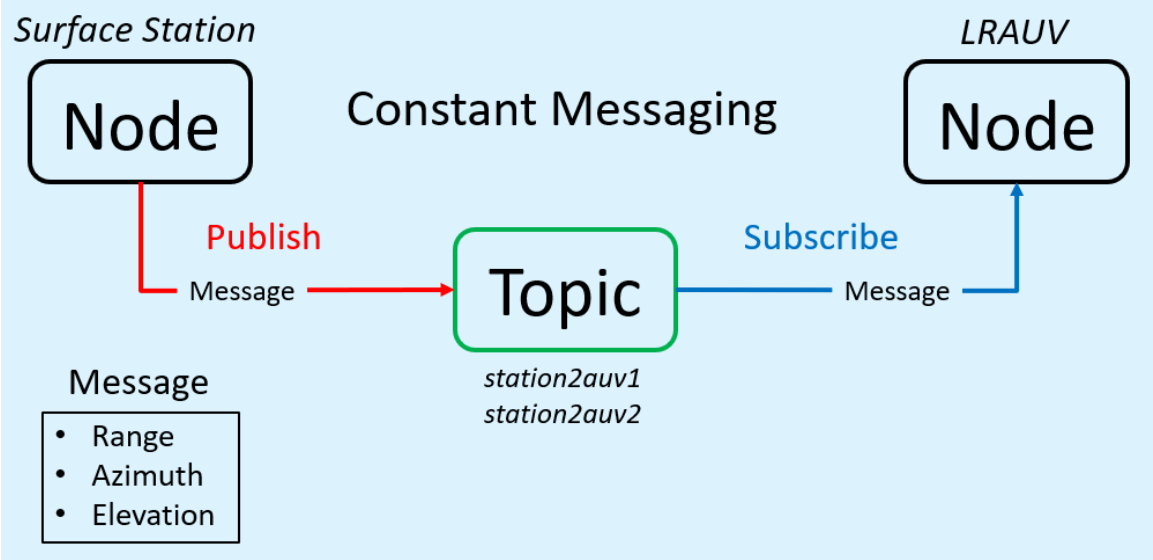

Figure 10: Communication Diagram for CANON simulation.

All of the communications implemented for this project is based off of the acoustic ultra-short baseline (USBL) communication used by the LRAUVs. At its core, a USBL system consists of a transceiver and a transponder. The transceiver pings the transponder, which replies with its own ping, which is then received by the transceiver and used to calculate the range and bearing of the transponder for positioning.

The communication implemented for this scenario utilizes topics, asynchronous 'channels' discussed earlier in this paper in the ROS section, with each LRAUV subscribing to its own unique topic being published to by the Surface Station ('*station2auv1/auv2*' in the diagram). In USBL fashion, the messages on the topic contain range, azimuth, and elevation from the Surface Station to the respective LRAUV, which is calculated by the Surface Station using robot model position information natively available through ROS and Gazebo. Communication at this stage is constant, due to the use of topics, and flawless, unlike with real acoustic communications, which is by nature unreliable and dramatically worsens with range.

Images of the resulting simulation are in the Results section.

Scenario 2 – Targeted Sampling: Oceanographic Hotspot Detection

Targeted sampling is a familiar concept here at MBARI. Since much of the research being conducted concerns the entirety of the vast oceans, the ability to pinpoint features and areas of interest and make efficient use of MBARI vessels and equipment, is extremely valuable and often necessary. Based off of this idea, this scenario simulates the discovery of a particle plume and its subsequent further exploration by a pair of collaborative LRAUVs.

This second scenario also makes use of a neat and useful passive turbulent plume generation feature in UUV Simulator. User controlled plume parameters include buoyancy flux and number of particles generated, among others, and in addition, the user controlled current modelled into the environment influences the plume dispersion. Particle dynamics take a Lagrangian random walk approach (M. M. M. Manhães et al., 2016). Appropriately, this scenario will also make use of the chemical concentration sensor available on the ECA A9 robot model in order to detect the plume by concentration readings from the sensor. An example plume is shown below, with the color representing the age of the particle in the simulation.

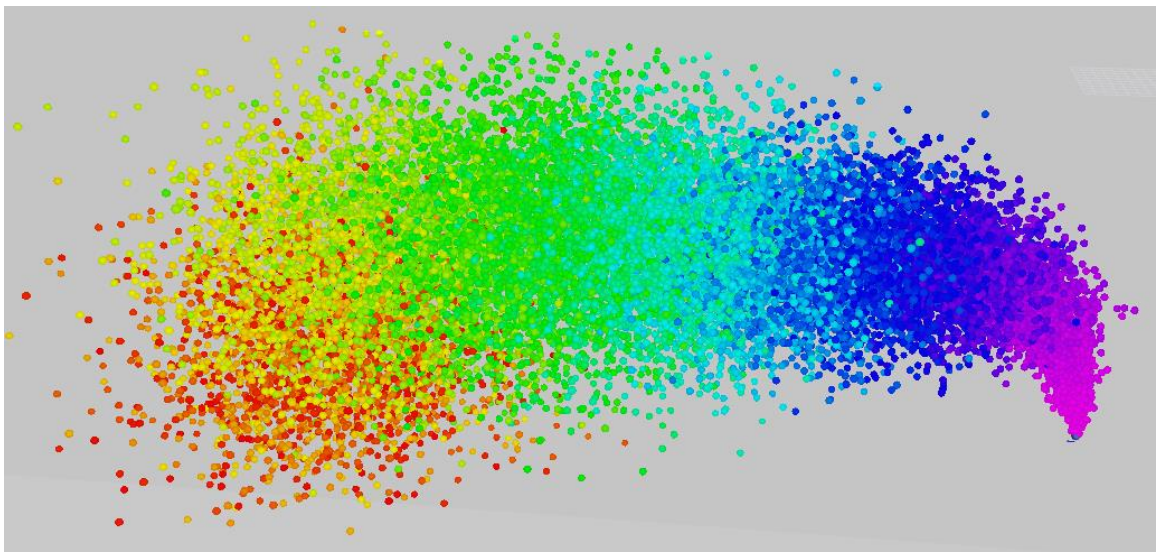

Figure 11: Particle plume in UUV Simulator.

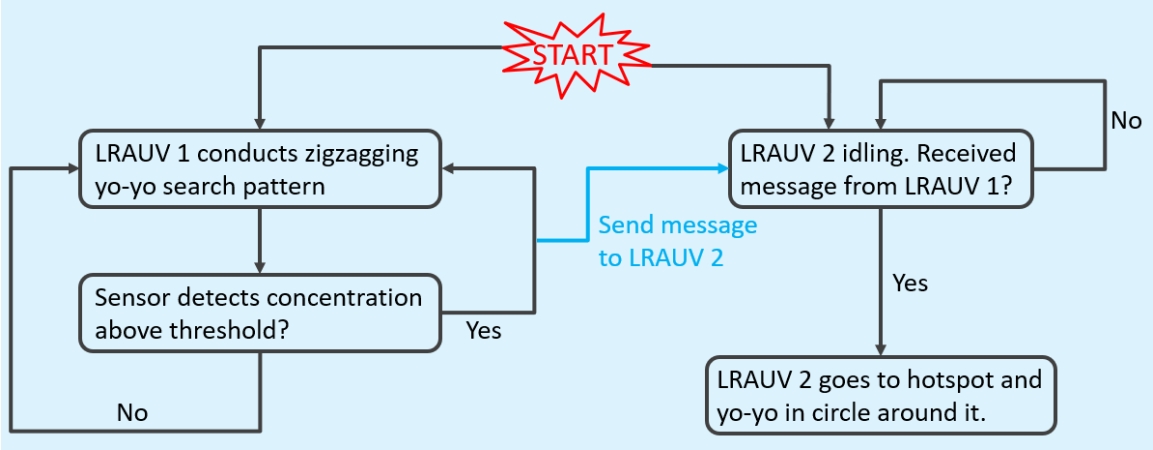

Figure 12: Breakdown of targeted sampling simulation in UUV Simulator.

In contrast to the CANON scenario, since there is no longer a Surface Station, all communication in this scenario is directly from one LRAUV to the other. To start, LRAUV 1 embarks on a broad, zigzagging movement incorporating the yo-yoing gait, searching for the plume, while LRAUV 2 simply idles at its starting point. Once LRAUV 1 senses a particle concentration greater than some programmed threshold, it sends a message to the idling LRAUV 2 and continues on its zigzagging search pattern. In a scaled up scenario, this would allow for LRAUV 1 to keep searching and potentially find more plumes/hotspots and relay them to LRAUV 2. When LRAUV 2 receives the message, it proceeds to the location conveyed in the message and performs a yo-yoing circular motion to further take measurements in the vicinity.

The communication has also been made more realistic – adding in randomized failure to represent acoustic unreliability and using a ROS service to change the communication from constant to trigger-based. As previous mentioned, a service in ROS implements the request/reply paradigm; sending a request message to the service will result in a reply message. But in between receiving the request and sending a reply, a service can do much more, limited only by the code written for the service's callback function. For this implementation, the service acts as a switch for the overall LRAUV-to-LRAUV communication. LRAUV 1 sends a request message to the '*send_measurement*' service when it senses a high enough particle concentration. This triggers the service to calculate the random failure, then publish a message containing the concentration and the x/y/z coordinates of the current location if rolled for successful message transmission. Note that this published message is **not** the service's reply message, which is sent after a

random failure or a published message and consists of a Boolean indicating False if the random failure occurred, or True if the message was published. The published message is then read by LRAUV 2 over the '*measurement_out*' topic, after which it heads to the location specified to begin further exploration. See diagram below for visual breakdown.

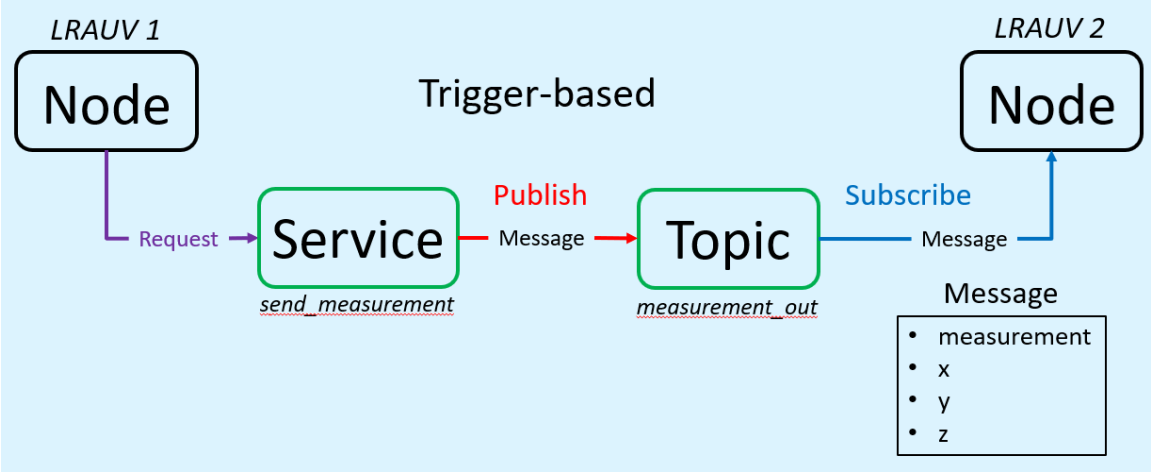

Figure 13: Communication Diagram for Targeted Sampling simulation.

Images of the resulting simulation are in the Results section.

RESULTS

Scenario 1 – CANON simulation

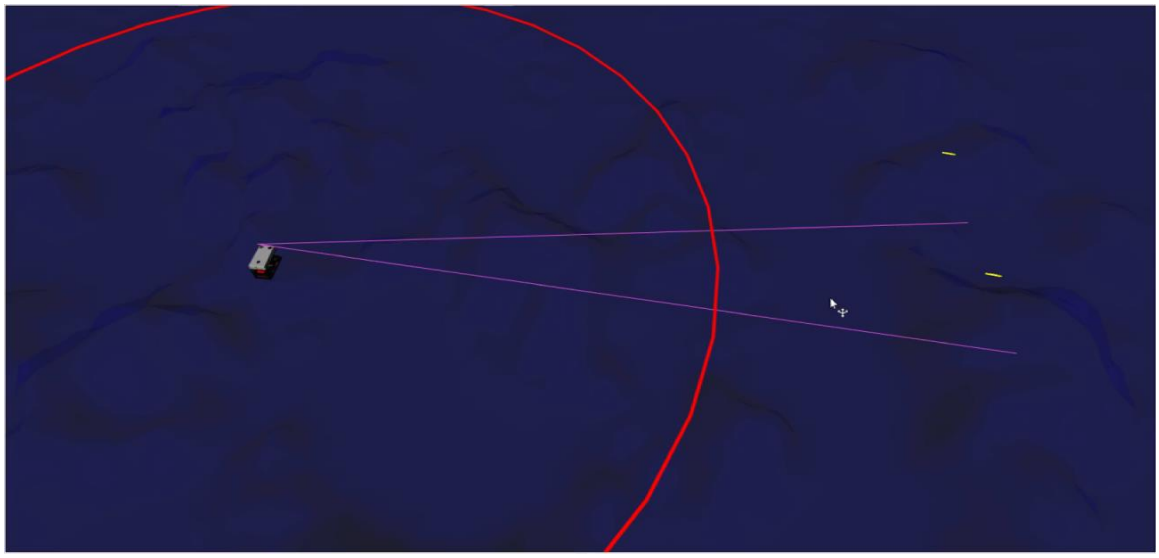

Figure 14: Starting positions for LRAUVs. ROV to the left represents the Surface Station.

Figure 15: LRAUVs have gotten close enough to the Surface Station to split off into their individual behaviors. Circular yo-yo movement LRAUV is splitting off while the other LRAUV continues to the Surface Station.

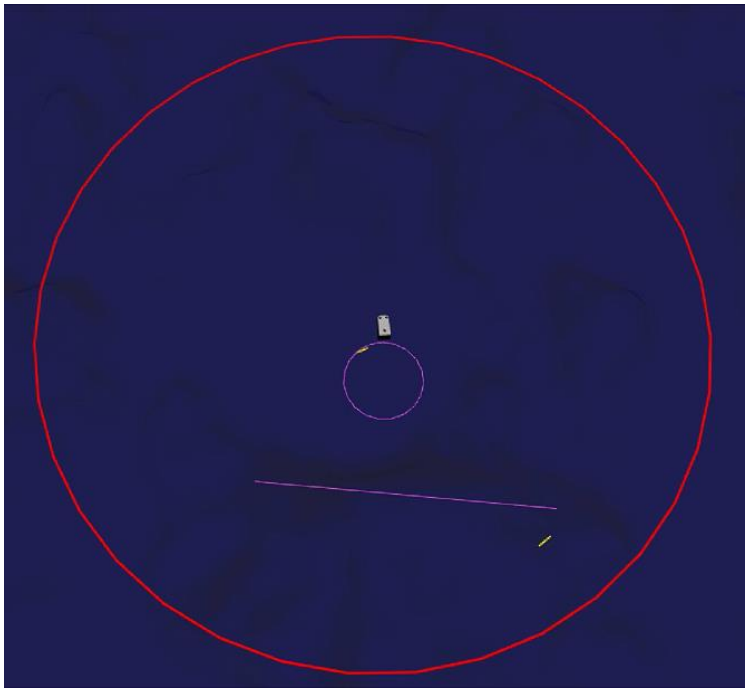

Figure 16: LRAUV has reached Surface Station and proceeds to idle. Other LRAUV is on next leg of circular gait.

The only problem with the simulation was that when the LRAUV reached the Surface Station's position and proceeded to idle, it became unstable and proceeded to literally spiral out of control. A strong possibility is that the circular idling trajectory commanded the robot to go to a position it could not conveniently reach (possibly leading to a kinematic singularity-esque situation) and therefore the model becoming unstable was a result of the controller attempting to compensate.

Scenario 2 – Targeted Sampling

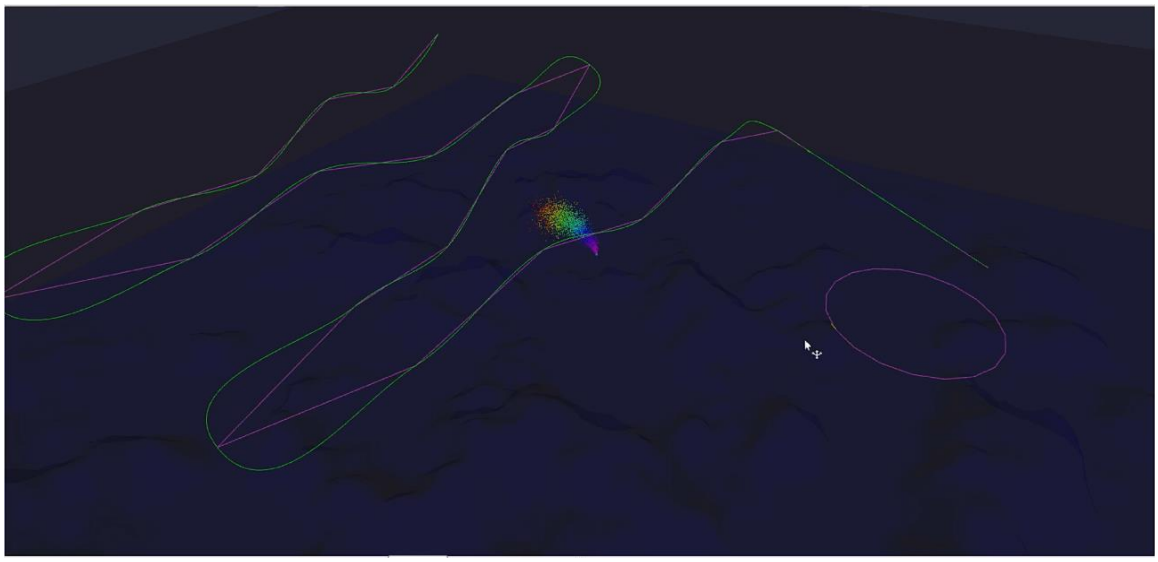

Figure 17: Start of targeted sampling simulation. LRAUV 1 has engaged zigzagging search pattern. LRAUV 2 is idling at its starting position. Plume can be seen in the center.

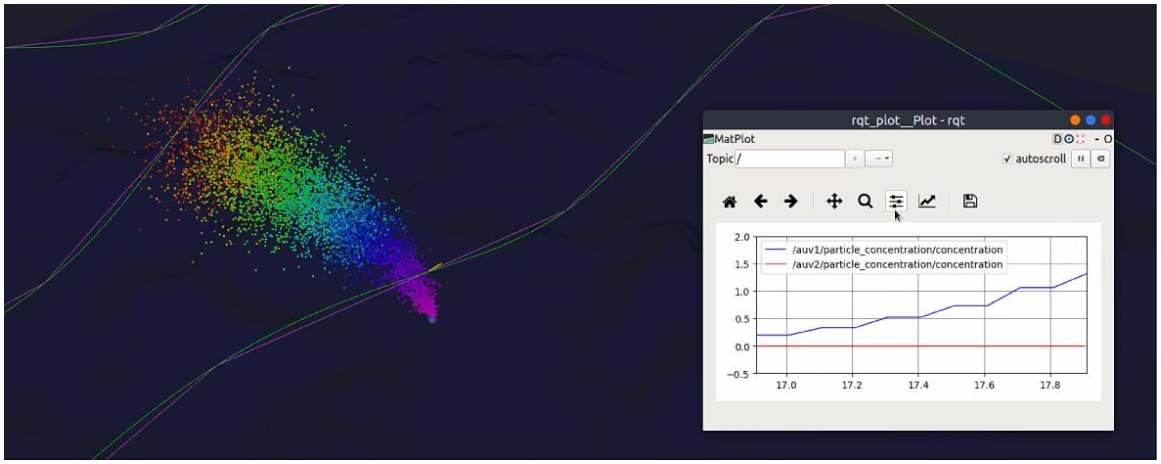

Figure 18: Searching LRAUV has encountered the plume and the live plot of the LRAUV's concentration readings from its sensor shows the high concentration that will trigger the message being sent to LRAUV 2.

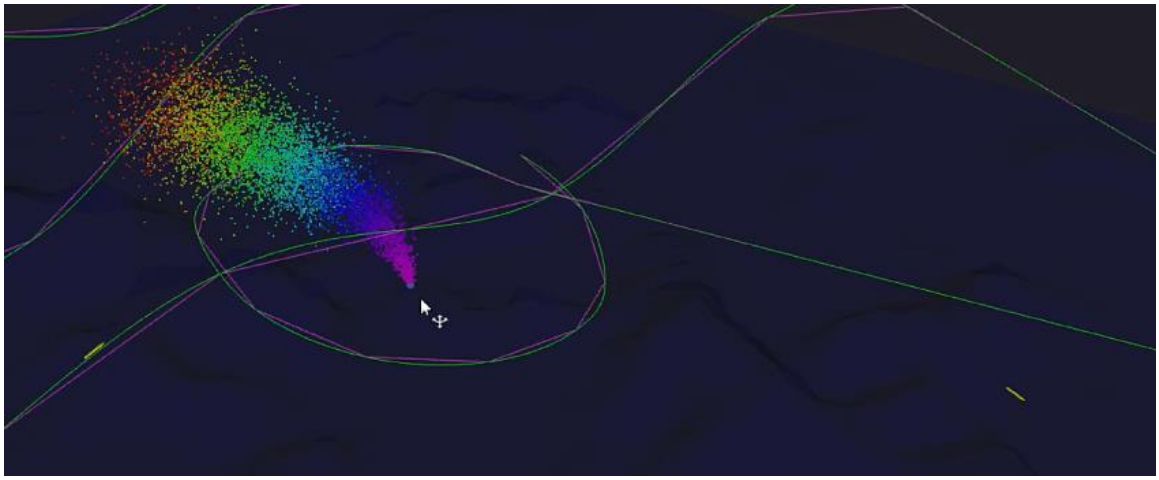

Figure 19: LRAUV 2's trajectory to the detection location shown. Searcher LRAUV continues on search pattern.

As mentioned previously, the LRAUV behaviors for this scenario could be very easily scaled for multiple plumes, provided that simulation of the robots sensing multiple plumes is doable (this has yet to be tested). All of the major behavioral and communication code would be essentially unchanged and only a few data structure and conditional logic changes would be required.

DISCUSSION

While the previous two simulations clearly show the usefulness of the UUV simulator's behavioral and environmental capabilities, it is important to note that these were performed with models of the ECA A9, and not the LRAUVs themselves. Fortunately, by leaving the physical mesh files of the model alone and parametrically modifying the various xml files that otherwise define a robot model in ROS, theoretically it is not too difficult to obtain a mimicry of an LRAUV in all but aesthetic. Furthermore, the LRAUV's underwater dynamics are defined in the current simulator, and similarly so to how the UUV Simulator defines underwater dynamics for the sample vehicles, including the ECA A9. Thus, this provides a natural starting point to creating a mimic LRAUV model out of the existing ECA A9 model.

The underwater dynamics are mostly based off of the following equation of motion:

$$(M_{RB} + M_A)\dot{v}_r + \big(C_{RB}(v_r) + C_A(v_r)\big)v_r + D(v_r)v_r + g(\eta) + g_0 = \tau$$

(Fossen, 1994)

The *RB* subscript terms account for the rigid-body inertial, coriolis, and centripetal dynamics; combined with the restoring forces, these are provided by Gazebo's physics engine. The remaining *A* subscript added mass terms and drag (*D*) are accounted for by the plugins from the UUV Simulator, which take into account various hydrostatic parameters, added mass coefficients, and drag coefficients defined in the xml files of a ROS robot model. The *v* term is the velocity vector, representing the vehicle's linear and roll/pitch/yaw velocity.

The some significant parameters that were transferred over from the LRAUV parameters in the current simulator to the existing ECA A9 model were: volume, length, width, center of buoyancy, quadratic damping coefficients, and added mass coefficients. Interestingly, the ECA A9 model only included linear damping coefficients, whereas the LRAUV only used quadratic damping; therefore the existing linear coefficients were supplanted with quadratic coefficients. The ECA A9 model was also quite thorough and came with a smattering of parameters and characteristics that had no counterpart defined in the current LRAUV simulator. These were left unmodified.

The modified, mimic LRAUV model was tested by uploading it into the targeted sampling scenario. While the model was stable – something not to take lightly – it was significantly more sluggish than the ECA A9 and had difficulty adhering to the trajectory laid out by the controller. It also seemed to have little or no sense of depth control. Unfortunately, due to time constraints there was no further iteration on this mimic model and no definitive conclusion can be reached about the performance or feasibility of this method until the modified hydrodynamic parameters are further tuned and/or the hydrodynamic calculations for both the UUV Simulator and the current LRAUV simulator compared and the model adjusted accordingly.

CONCLUSIONS/RECOMMENDATIONS

This paper has shown the ability of ROS and the Unmanned Underwater Vehicle Simulator (UUV Simulator) to provide an informative and engaging visual of MBARI's LRAUVs in action. Through Gazebo, the simulation provides a glimpse of the robots in a realistic and customizable marine environment, while Rviz provides a visualization of

environmental data, its interaction with the robots, and the subsequent influence on the robots' behavior. Furthermore, the ease with which additional robots or other objects can be added in simulation via Gazebo/Rviz and in software as ROS nodes, combined with ROS's accessible inter-node communications, allows for convenient simulation of complex experimental scenarios, such as portions of the Spring 2019 CANON experiment, operations of many LRAUVs simultaneously, or those involving many different MBARI assets.

Although there is a strong foundation for LRAUV simulation with what ROS and UUV Simulator currently offer, especially considering the work done over just ten weeks for this project, they can be improved in a few ways to suit MBARI's needs. First, the robot models that have been successfully used for simulation thus far are physically and hydrodynamically different compared to the LRAUV. This can be remedied in two ways: continuing the modification of the ECA A9 robot model to create a mimicry of an LRAUV, or creating an entirely new robot model, which would entail more control over the model at the cost of requiring more work. Once a satisfactory robot model is available, a logical next step would be to write the software, providing the LRAUV model's interface to ROS as well as the algorithmic and behavioral code. Lastly, should it be of interest, models of other MBARI assets, such as the AUVs, ROVs, Wave Gliders, and moorings, can be created for use in simulation with the LRAUVs.

ACKNOWLEDGEMENTS

I would like to thank my mentors, Yanwu Zhang and Ben Raanan, for not only providing all of the advice, reference material, and guidance that I needed to succeed with my project, but also going the extra mile to ensure that my time here was both valuable and memorable.

Also, many thanks to our internship coordinator, George Matsumoto, and intern logistics coordinator, Maddie Heard, for organizing this program, taking care of us and meeting our every need, and coordinating all of the fantastic tours, outings, and other social occasions that enriched our time here and brought us together.

To my fellow interns, as a Midwestern engineer in a pack of seafarers and biologists, thank you for introducing me to a lifestyle that I never would have seen in Chicago, yet is undoubtedly second nature to many of you, and putting up with me while I stumbled my way through it.

Last but certainly not least, my profound gratitude to the Monterey Bay Aquarium Research Institute and the David and Lucile Packard Foundation for making this all possible and bringing enlightenment and an unforgettable experience to people like me.

References:

Brown, K. (2019, May 29). *Spring 2019 CANON experiment explores Earth's largest migration*. Retrieved from Monterey Bay Aquarium Research Institute: https://www.mbari.org/canon-spring-2019/

Fossen, T. I. (1994). *Guidance and Control of Ocean Vehicles.* John Wiley & Sons.

Google. (2019, August 12). *Quigley: ROS: an open-source Robot Operating System - Google Scholar*. Retrieved from Google Scholar: https://scholar.google.com/scholar?cites=14376749257557382 6&as_sdt=2005&sciodt=0,5&hl=en

Manhaes, M. M., Scherer, S. A., Voss, M., Douat, L. R., & Rauschenbach, T. (2016). UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation. *OCEANS 2016 MTS/IEEE Monterey.* Monterey: IEEE. doi:10.1109/oceans.2016.7761080

Monterey Bay Aquarium Research Institute. (2018, August 22). *Long-range autonomous underwater vehicle Tethys*. Retrieved from Monterey Bay Aquarium Research Institute Website: https://www.mbari.org/at-sea/vehicles/autonomous-underwater-vehicles/long-range-auv-tethys/

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., . . . Ng, A. (2009). ROS: an open-source Robot Operating System. *ICRA Workshop on Open Source Software*.